SPD domain-specific batch normalization to crack interpretable unsupervised domain adaptation in EEG

Reinmar J. Kobler$^{1,2}$, Jun-ichiro Hirayama$^1$, Qibin Zhao$^1$, Motoaki Kawanabe$^{1,2}$
$^1$RIKEN Center for Advanced Intelligence Project (RIKEN AIP), Tokyo, Japan
$^2$Advanced Telecommunications Research Institute International (ATR), Kyoto, Japan
kobler.reinmar@gmail.com, qibin.zhao@riken.jp
jun-ichiro.hirayama@a.riken.jp
kawanabe@atr.jp

Abstract
Electroencephalography (EEG) provides access to neuronal dynamics non-invasively with millisecond resolution, rendering it a viable method in neuroscience and healthcare. However, its utility is limited as current EEG technology does not generalize well across domains (i.e., sessions and subjects) without expensive supervised re-calibration. Contemporary methods cast this transfer learning (TL) problem as a multi-source/-target unsupervised domain adaptation (UDA) problem and address it with deep learning or shallow, Riemannian geometry aware alignment methods. Both directions have, so far, failed to consistently close the performance gap to state-of-the-art domain-specific methods based on tangent space mapping (TSM) on the symmetric, positive definite (SPD) manifold. Here, we propose a machine learning framework that enables, for the first time, learning domain-invariant TSM models in an end-to-end fashion. To achieve this, we propose a new building block for geometric deep learning, which we denote SPD domain-specific momentum batch normalization (SPDDSMBN). A SPDDSMBN layer can transform domain-specific SPD inputs into domain-invariant SPD outputs, and can be readily applied to multi-source/-target and online UDA scenarios. In extensive experiments with 6 diverse EEG brain-computer interface (BCI) datasets, we obtain state-of-the-art performance in inter-session and -subject TL with a simple, intrinsically interpretable network architecture, which we denote TSMNet.
Code: https://github.com/rkobler/TSMNet

1 Introduction
Electroencephalography (EEG) measures multi-channel electric brain activity from the human scalp with millisecond precision [1]. Transient modulations in the rhythmic brain activity can reveal cognitive processes [2], affective states [3] and a person’s health status [4]. Unfortunately, these modulations exhibit low signal-to-noise ratio (SNR), domain shifts (i.e., changes in the data distribution) and have low specificity, rendering statistical learning a challenging task - particularly in the context of brain-computer interfaces (BCI) [5] where the goal is to predict a target from a short segment of multi-channel EEG data in real-time.
Under domain shifts, domain adaptation (DA), defined as learning a model from a source domain that performs well on a related target domain, offers principled statistical learning approaches with theoretical guarantees [6, 7]. DA in the BCI field mainly distinguishes inter-session and -subject transfer learning (TL) [8]. In inter-session TL, domain shifts, are expected across sessions mainly due to mental drifts (low specificity) as well as differences in the relative positioning of the electrodes and their impedances. Inter-subject TL is more difficult, as domain shifts are additionally driven by structural and functional differences in brain networks as well as variations in the performed task [9].
These domain shifts are traditionally circumvented by recording labeled calibration data and fitting domain-specific models [10, 11]. As recording calibration data is costly, models that are robust to
scarce data with low SNR perform well in practice. Currently, tangent space mapping (TSM) models \cite{ehrsperger2022deep,becker2022unsupervised} operating with symmetric, positive definite (SPD) covariance descriptors of preprocessed data are considered state-of-the-art (SoA) \cite{li2019unsupervised, ju2020geometric}. They are well suited for EEG data as they exhibit invariances to linear mixing of latent sources \cite{ek2010time}, and are consistent \cite{herve2019estimation} and intrinsically interpretable \cite{li2019unsupervised} estimators for generative models that encode label information with a log-linear relationship in source power modulations.

Competitive, supervised calibration-free methods are one of the long-lasting grand challenges in EEG neurotechnology research \cite{kim2015unsupervised, ju2020neural, ju2020convolutional, ju2020unsupervised}. Among the applied transfer learning techniques, including multi-task learning \cite{soltani2017multi} and domain-invariant learning \cite{gerez2018domain, guan2019domain, guan2020domain}, unsupervised domain adaptation (UDA) \cite{long2015learning} is considered as key to overcome this challenge \cite{li2019unsupervised, ju2020neural}. Contemporary methods cast the problem as a multi source and target UDA problem and address it with deep learning \cite{saha2017learning, dai2017unsupervised} or shallow, Riemannian geometry aware alignment methods \cite{ju2018geometric, ju2019unsupervised}. Successful methods must cope with notoriously small and heterogeneous datasets (i.e., dozens of domains with a few dozens observations per domain and class). In a recent, relatively large scale inter-subject and -dataset TL competition with few labeled examples per target domain \cite{ju2020neural}, deep learning approaches that aligned the first and second order statistics either in input \cite{boult2022unsupervised, sachtler2022unsupervised} or latent space \cite{ju2020unsupervised} obtained the highest scores. Whereas, in a pure UDA competition \cite{ju2019unsupervised} with a smaller dataset, Riemannian geometry aware approaches dominated. With the increasing popularity of geometric deep learning \cite{ju2019unsupervised}, Ju and Guan \cite{ju2020un} proposed an architecture based on SPD neural networks \cite{li2019unsupervised} to align SPD features in latent space and attained SoA scores. Despite the tremendous advances in recent years, the field still lacks methods that can consistently close the performance gap to state-of-the-art domain-specific methods.

To close this gap, we propose a machine learning framework around domain-specific batch normalization on the SPD manifold (Figure 1). The proposed framework is used to implement domain-specific TSM (Figure 1a), which requires tracking the domains’ Fréchet means in latent space as they are changing during training a typical TSM model in an end-to-end fashion (Figure 1b). After reviewing some preliminaries in section 2, we extend momentum batch normalization (MBN) \cite{chen2020large} to SPDMBN that controls the Fréchet mean and variance of SPD data in section 3. In a theoretical analysis, we show under reasonable assumptions that SPDMBN can track and converge to the data’s true Fréchet mean, enabling, for the first time, end-to-end learning of feature extractors, TSM and tangent space classifiers. Building upon this insight, we combine SPDMBN with domain-specific batch normalization (DSBN) \cite{ju2020geometric} to form SPDDSMBN (Figure 1a). A SPDDSMBN layer can transform domain-specific SPD inputs into domain-invariant SPD outputs (Figure 1c). Like DSBN, SPDDSMBN easily extends to multi-source, multi-target and online UDA scenarios. In section 4, we briefly review the generative model of EEG, before the proposed methods are combined in a simple, intrinsically interpretable network architecture, denoted TSMNet (Figure 1b). We obtain state-of-the-art performance in inter-session and -subject UDA on small and large scale EEG BCI datasets, and show in an ablation study that the performance increase is primarily driven by performing DSBN on the SPD manifold.

## 2 Preliminaries

**Multi-source multi-target unsupervised domain adaptation**

Let $\mathcal{X}$ denote the space of input features, $\mathcal{Y}$ a label space, and $\mathcal{I}_d \subset \mathbb{N}$ an index set that contains unique domain identifiers. In
While, parallel transport is generally defined for tangent space vectors (i.e., the space of real symmetric matrices). To shift a set of tangent space points to vary around a parametrized mean $\mu$, we can use parallel transport. Tangent spaces have Euclidean structure with easy to compute distances due to affine invariance. The Fréchet variance $\nu_Z^2$ is defined as the attained value at the minimizer $G_Z$:

$$\nu_Z^2 = \text{Var}_Z(G_Z) = \frac{1}{M} \sum_{j=1}^{M} \delta_{AIRM}(G_Z, Z_j)$$

To shift a set of tangent space points to vary around a parametrized mean $G_\phi$, parallel transport on $S_D^+$ can be used:

$$\Gamma_{G_Z \rightarrow G_\phi}(S) = E^T S E, \quad E = (G_Z^{-1} G_\phi)^{\frac{1}{2}}$$

While, parallel transport is generally defined for tangent space vectors $S$, on $S_D^+$ the same operations also apply directly to points on the manifold (i.e., $Z \in Z$) for ease of notation, although not required by our method, we assume that $M$ is the same for each domain.

For SPD matrices, powers, logarithms and exponentials can be computed via eigen decomposition.
Table 1: Overview and differences of relevant batch normalization algorithms. The last column, denoted normalization, summarizes which statistics are used to normalize the batch data during training.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>$S_D^+$</th>
<th>domain-specific</th>
<th>momentum $\gamma$</th>
<th>normalization</th>
</tr>
</thead>
<tbody>
<tr>
<td>MBN [32]</td>
<td>no</td>
<td>no</td>
<td>adaptive</td>
<td>running stats</td>
</tr>
<tr>
<td>SPDDBN [46]</td>
<td>yes</td>
<td>no</td>
<td>fixed</td>
<td>running stats</td>
</tr>
<tr>
<td>SPDMBN (algorithm) proposed</td>
<td>yes</td>
<td>no</td>
<td>adaptive</td>
<td>running stats</td>
</tr>
<tr>
<td>DSBN [38]</td>
<td>no</td>
<td>yes</td>
<td>fixed</td>
<td>batch stats</td>
</tr>
<tr>
<td>SPDDSMBN [48] proposed</td>
<td>yes</td>
<td>yes</td>
<td>adaptive</td>
<td>running stats</td>
</tr>
</tbody>
</table>

3 Domain-specific batch normalization on $S_D^+$

In this section, we review relevant batch normalization (BN) [45] variants with a focus on $S_D^+$. We then present SPDMBN and show in a theoretical analysis that the running estimate converges to the true Fréchet mean under reasonable assumptions. At last, we combine the idea of domain-specific batch normalization (DSBN) [38] with SPDMBN to form a SPDDSMBN layer. Table 1 provides a brief overview of related and proposed methods.

**Batch normalization** Batch normalization (BN) [45] is a widely used training technique in deep learning as BN layers speed up convergence and improve generalization via smoothing of the energy landscape [47] [32]. A standard BN layer applies slightly different transformations during training and testing to independent and identically distributed (iid) observations $x_j \in \mathbb{R}^d$ within the $k$-th minibatch $B_k$ of size $M$ drawn from a dataset $T$. During training, the data are normalized using the batch mean $\mu_k$ and variance $\sigma^2_k$, and then scaled and shifted to have a parametrized mean $\mu_\phi$ and variance $\sigma^2_\phi$. Internally, the layer updates running estimates of the dataset’s statistics ($\mu_k, \sigma^2_k$) during each training step $k$; the updates are computed via exponential smoothing with momentum parameter $\gamma$. During testing, the running estimates are used.

Using batch statistics to normalize data during training rather than running estimates introduces noise whose level depends on the batch size [32]: smaller batch sizes raise the noise level. The introduced noise regularizes the training process, which can help to escape poor local minima in initial learning but also lead to underfitting. Momentum BN (MBN) [32] allows small batch sizes while avoiding underfitting. Like batch renormalization [48], MBN uses running estimates during training and testing. The key difference is that MBN keeps two sets of running statistics; one for training and one for testing. The latter are updated conventionally, while the former are updated with momentum parameter $\gamma_{\text{train}}(k)$ that decays over training steps $k$. MBN can, therefore, quickly escape poor local minima during initial learning and avoid underfitting at later stages [32].

**Batch normalization on $S_D^+$** It is intractable to compute the Fréchet mean $G_{B_k}$ for each minibatch $B_k = \{Z_j \in S_D^+\}_{j=1}^M$, as there is no efficient algorithm to solve [4]. Brooks et al. [44] proposed Riemannian Batch Normalization (RBN) as a tractable approximation. RBN approximateley solves [4] by aborting the iterative Karcher flow algorithm after one iteration. To transform $Z_j \in B_k$ with estimated mean $\mu_k$ to vary around $G_\phi$, parallel transport [7] is used. The RBN input output transformation is then expressed as

$$\text{RBN}(Z_j; G_\phi, \gamma) = \Gamma_{B_k \rightarrow G_\phi}(Z_j) = E^T Z_j E, \quad E = (B_k^{-1} G_\phi)^{\frac{1}{2}}, \quad \forall Z_j \in B_k \quad (8)$$

Using [40], the running estimate of the dataset’s Fréchet mean can be updated in closed form

$$G_k = G_{k-1} \#_{\gamma} B_k \quad (9)$$

In [40] we proposed an extension to RBN, denoted SPD batch renormalization (SPDBN) that controls both Fréchet mean and variance. Like batch renormalization [48], SPDBN uses running estimates $G_k$ and $\nu^2_k$ during training and testing. To transform $Z_j \in B_k$ to vary around $G_\phi$ with variance $\nu^2_\phi$, each observation is first transported to vary around the identity matrix $I$, rescaled via computing matrix powers and finally transported to vary around $G_\phi$. The sequence of operations can be expressed as

$$\text{SPDBN}(Z_j; G_\phi, \nu^2_\phi, \varepsilon, \gamma) = \Gamma_1 \rightarrow G_\phi \circ \Gamma_{G_\phi \rightarrow I}(Z_j) \nu^{\frac{\nu_\phi}{\nu_\phi + \varepsilon}}, \quad \forall Z_j \in B_k \quad (10)$$

The standard backpropagation framework with extensions for structured matrices [49] and manifold-constrained gradients [40] can be used to propagate gradients through RBN and SPDBN layers and learn the parameters $(G_\phi, \nu_\phi)$. 


We assume that the variance

We define a dataset that contains the latent representations generated with feature set

As MBN \[32\], we decay

Algorithm 1: SPD momentum batch normalization (SPDMBN)

Input:

\( \text{Output:} \) normalized batch \( \{ Z_j \} = \text{SPDMBN}(Z_j) \in S^+_D \mid Z_j \in B_k \}

if training then

\( B_k = \text{karcher\_flow}(B_k, \text{steps} = 1) \);  // approx. batch Fréchet mean

\( G_k = G_{k-1} \# \gamma_{\text{train}}(k) B_k \);  // update running stats

\( \bar{v}^2_k = (1 - \gamma_{\text{train}}(k)) \bar{v}^2_{k-1} + \gamma_{\text{train}}(k) \text{Var}_{\theta_k}(G_k) \)

\( \tilde{G}_k = G_{k-1} \# \tilde{\mathbb{B}}_k \)

\( \bar{v}^2_k = (1 - \gamma) \bar{v}^2_{k-1} + \gamma \text{Var}_{\theta_k}(G_k) \)

end

\( (G_k, \nu^2_k) = (G_k, \nu_\theta) \) if training else \( (\tilde{G}_k, \nu^2_k) \)

\( Z_j = \Gamma_{E \to G_\theta} \circ \Gamma_{G_k \to 1}(Z_j)^\gamma_{\text{train}} + \gamma \)  // parallel transport to whiten, rescale and rebias

Momentum batch normalization on \( \mathbb{S}^+_D \) SPDBN \[43\] suffers from the same limitations as batch renormalization \[43\]. Consequently, we propose to extend MBN \[12\] to \( \mathbb{S}^+_D \). We list the pseudocode of our proposed extension, which we denote SPDMBN, in algorithm \[1\]. SPDDBN uses approximations of batch-specific Fréchet means to update two sets of running estimates of the dataset’s Fréchet mean. As MBN \[12\], we decay \( \gamma_{\text{train}}(k) \) with a clamped exponential decay schedule

\[
\gamma_{\text{train}}(k) = 1 - \gamma_{\text{min}} \text{max}(K - k, 0) + \gamma_{\text{min}} \quad (11)
\]

where \( K \) defines the training step at which \( \gamma_{\text{min}} \in [0, 1] \) should be attained.

The running mean in SPDMBN converges to the Fréchet mean Here, we consider models that apply a SPDMBN layer to latent representations generated by a feature extractor \( f_\theta : \mathcal{X} \to \mathbb{S}^+_D \) with learnable parameters \( \theta \).

We define a dataset that contains the latent representations generated with feature set \( \theta_k \) as \( Z_{\theta_k} = \{ f_{\theta_k}(x) \mid x \in T \} \), and a minibatch of \( M \) iid samples at training step \( k \) as \( B_k \). We denote the Fréchet mean of \( Z_{\theta_k} \) as \( G_{\theta_k} \), the estimated Fréchet mean, defined in \( \theta \), as \( G_k \), and the estimated batch mean as \( B_k \). Since the batches are drawn randomly, we consider the batch and running means as random variables.

We assume that the variance \( \text{Var}_{\theta_k}(B_{k-1}) = \mathbb{E}_{B_{k-1}}[\delta^2(B_{k-1}, G_{\theta_k})] \) of the previous batch mean \( B_{k-1} \) with respect to the current Fréchet mean \( G_{\theta_k} \) is bounded by the current variance \( \text{Var}_{\theta_k}(B_k) \) and the norm of the difference in the parameters

\[
\text{Var}_{\theta_k}(B_{k-1}) \leq (1 + ||\theta_k - \theta_{k-1}||) \text{Var}_{\theta_k}(B_k) \quad (12)
\]

That is, across training steps \( k \) the parameter updates are required to change the first and second order moments of the distribution of \( Z_{\theta_k} \) gradually so that the expected distance between \( G_{\theta_k} \) and the previous batch mean \( B_{k-1} \) is bounded. We conjecture that this is the case for feature extractors \( f_\theta \) that are smooth in the parameters and small learning rates, but leave the proof for future work.

**Proposition 1** (Error bound for \( G_k \)). Consider the setting defined above, and assumption \( 12 \) holds true. Then, the variance of the running mean \( \text{Var}_{\theta_k}(G_k) \) is bounded by

\[
\text{Var}_{\theta_k}(G_k) \leq \text{Var}_{\theta_k}(B_k) \quad (13)
\]

over training steps \( k \) if

\[
||\theta_k - \theta_{k-1}|| \leq \frac{1 - \gamma^2}{(1 - \gamma)^2} - 1 \quad (14)
\]

holds true.

The proof is provided in appendix \([A.1]\) of the supplementary material and relies on the proof of the geometric law of large numbers \([53]\).

**Proposition** \( 1 \) states that if \( 12 \) and \( 14 \) are met, the expected distance between the true Fréchet mean and the running mean is less or equal to the one of the batch mean. Consequently, the introduced
We propose to use a SPDMBN layer followed by a LogEig layer. A DSBN layer mapping (LERM) projects features to the tangent space at the identity matrix. As a result, distances between LogEig invariance properties (e.g., affine mixing) projects features to an Euclidean vector space where standard layers can be applied and distances are cheap to compute.

Since $\gamma$ controls the adaptation speed of $G_k$, proposition also states that if $\gamma$ converges to zero (no adaptation), the parameter updates are required to converge to zero as well (no learning). Hence, for a fixed $\gamma \in (0, 1)$, as in the case of SPDBN, proposition is fulfilled, if the learning rate for the parameters $\theta$ is chosen sufficiently small. This can substantially slow down initial learning for standard choices of $\gamma$ (e.g., 0.1 or 0.01).

As a remedy, SPDMBN uses an adaptive momentum parameter, which allows larger parameter updates during initial training steps. If we consider a late stage of learning, and in particular assume that after a certain number of iterations $k$ the parameters stay in a small ball with radius $\rho$ around $\theta^*$ (i.e., $\|\theta_k - \theta^*\| \leq \rho$ $\forall k > \kappa$) and the feature extractor is $L$-smooth in the parameters (i.e., $\delta(f_{\theta}(x), f_{\theta}(\tilde{x})) \leq L||\theta - \tilde{\theta}|| \forall x \in T$, $\forall \theta, \tilde{\theta}$) then the distances are bounded $\delta(f_{\theta_k}(x), f_{\theta_k}(\tilde{x})) \leq \rho L$.

**Remark 1** (Convergence of $G_k$ for SPDMBN). If $\rho L$ is negligible compared to the dataset’s variance, then the Fréchet mean and variance can be considered fixed, and the theorem of large numbers on $S^+_D$ [50] applies directly. That is, if the momentum parameter is decayed exponentially $\forall k > \kappa$ the running mean $G_k$ converges to the Fréchet mean $G_{\theta^*}$ in probability as $k \to \infty$.

Taken together, Proposition and Remark provide guidelines to update $G_k$ in SPDMBN so that the introduced estimation error is bounded during initial fast learning (large $\gamma$) and decays towards zero in late learning (small $\gamma$).

**SPDMBN to learn tangent space mapping at Fréchet means** Typical TSM models for classification and regression first use $G_k$ to project $Z \in T \subset S^+_D$ to the tangent space at the Fréchet mean $G_T$, then use $G_k$ to transport the result to vary around $I$, and finally extract elements in the upper triangular part to reduce feature redundancy. The invertible mapping $P_{G_T} : S^+_D \to \mathbb{R}^{D(D+1)/2}$ is expressed as:

$$P_{G_T}(Z) = upper \circ \Gamma_{G_T}^{-1} \circ Log_{G_T}(Z) = upper(log(G_T^{\frac{1}{2}}ZG_T^{-\frac{1}{2}}))$$ (15)

We propose to use a SPDMBN layer followed by a LogEig layer to compute a similar mapping $m_\phi$ (Figure 4). A LogEig layer simply computes the matrix logarithm and vectorizes the result so that the norm is preserved. If the parametrized mean of SPDMBN is fixed to the identity matrix ($G_\phi = I$), the composition computes

$$m_\phi(Z) = LogEig \circ SPDMBN(Z) = upper \circ \log \circ \Gamma_{G_k}^{-1}(Z)^{\nu_k}$$

$$= upper \left( \frac{\nu_k}{\nu_k + \varepsilon} \log \left( G_k^{\frac{1}{2}}ZG_k^{-\frac{1}{2}} \right) \right)$$ (16)

where $(G_k, \nu_k^2)$ are the estimated Fréchet mean and variance of the dataset $T$ at training step $k$, and $\phi = \{\nu_k\}$ the learnable parameters. According to remark $G_k$ converges to $G_T$ and, in turn, $m_\phi$ to a scaled version of $P_{G_T}$, since upper is linear.

The mapping $m_\phi$ offers several advantageous properties. First, the features are projected to a Euclidean vector space where standard layers can be applied and distances are cheap to compute. Second, distances between the projected features locally approximate $\delta$ and, therefore, inherit its invariance properties (e.g., affine mixing) [41]. This improves upon a LogEig layer which projects features to the tangent space at the identity matrix. As a result, distances between LogEig projected features correspond to distances measured with the log-Euclidean Riemannian metric (LERM) which is not invariant to affine mixing. Third, controlling the Fréchet variance in empirically speeds up learning and improves generalization [46].

**Domain-specific batch normalization on $S^+_D$** Considering a multi-source UDA scenario, Chang et al. [58] proposed a domain-specific BN (DSBN) layer which simply keeps multiple parallel BN minibatches $B_i$ that form the union of $N_{B_k} \leq |T_d|$ domain-specific minibatches $B_k$ drawn from distinct domains $i \in I_{B_k} \subseteq I_d$. As before, each $B_k$ contains $j = 1, \ldots, M/N_{B_k}$ iid observations $x_j$. A DSBN layer mapping $\mathbb{R}^d \times I_d \to \mathbb{R}^d$ can then be expressed as

$$DSBN(x_j, i) = BN_{i}(x_j; g_\phi, s_\phi, \varepsilon, \gamma), \quad \forall x_j \in B_k^i, \quad \forall i \in I_{B_k}$$ (17)

To preserve the norm, the off diagonal elements are scaled by $\sqrt{Z}$. 

---

\[^{\text{37}}\text{To preserve the norm, the off diagonal elements are scaled by } \sqrt{Z}\]
In practice, the batch size $M$ is typically fixed. The particular choice is influenced by resource availability and the desired noise level introduced by minibatch based stochastic gradient descent. A drawback of DSBN is that for a fixed batch size $M$ and an increasing number of source domains $N_{D_k}$, the effective batch size declines for the BN layers within DSBN. Since small batch sizes increase the noise level introduced by BN, increasing the number of domains per batch can lead to underfitting. To alleviate this effect, we use the previously introduced SPDMBN layer. The proposed domain-specific BN layer on $S^+_D$ is then formally defined as

$$\text{SPDDM}_{\text{MBN}}(Z_j, i) = \text{SPDM}_{\text{MBN}}(Z_j; \mathbf{G}_\phi, \nu, \varepsilon, \gamma, \gamma_{\text{train}}(k)), \forall Z_j \in B_k^i \subset S^+_D, \forall i \in I_{B_k}$$

(18)

The layer can be readily adapted to new domains, as new SPDMBN layers can be added on the fly. If the entire data of a domain becomes available, the domain-specific Fréchet mean and variance can be estimated by solving (4), otherwise, the update rules in algorithm 1 can be used.

4 SPDDM to crack interpretable multi-source/-target UDA for EEG data

With SPDDM introduced in the previous section, we focus on a specific application domain, namely, multi-source/-target UDA for EEG-based BCIs and propose an intrinsically interpretable architecture which we denote TSMNet.

Generative model of EEG EEG signals $x(t) \in \mathbb{R}^P$ capture voltage fluctuations on $P$ channels. An EEG record (=domain) is uniquely identified by a subject and session identifier. After standard pre-processing steps, each domain $i$ contains $j = 1, ..., M$ labeled observations with features $X_{ij} \in \mathcal{X} \subset \mathbb{R}^{P \times T}$ where $T$ is the number of temporal samples. Due to linearity of Maxwell’s equations and Ohmic conductivity of tissue layers in the frequency ranges relevant for EEG [52], a domain-specific linear instantaneous mixture of sources model is a valid generative model:

$$X_{ij} = A_i S_{ij} + N_{ij}$$

(19)

where $S_{ij} \in \mathbb{R}^{Q \times T}$ represents the activity of $Q$ latent sources, $A_i \in \mathbb{R}^{P \times Q}$ a domain-specific mixing matrix and $N_{ij} \in \mathbb{R}^{P \times T}$ additive noise. Both $A_i$ and $S_{ij}$ are unknown which demands making assumptions on $A_i$ (e.g., anatomical prior knowledge [53]) and/or $S_{ij}$ (e.g., statistical independence [54]) to extract interesting sources.

Interpretable multi-source/-target UDA for EEG data As label information is available for the source domains, our goal is to identify discriminative oscillatory sources shared across domains. Our approach relies on TSM models with linear classifiers [12], as they are consistent [13] and intrinsically interpretable [17] estimators for generative models with log-linear relationships between the target $y_{ij}$ and variance $\text{Var}(s_{ij}^{(k)}(t))$ of $k = 1, ..., K \leq Q$ discriminative sources:

$$y_{ij} = \sum_{k=1}^{K} b_k \log \left( \text{Var}(s_{ij}^{(k)}(t)) \right) + \varepsilon_{ij}$$

(20)

where $b_k \in \mathbb{R}$ summarizes the coupling between the target $y_{ij}$ and the variance of the encoding source, and $\varepsilon_{ij}$ additive noise. In [17] we showed that the encoding sources’ coupling and their patterns (columns of $A_i$) can be recovered via solving a generalized eigenvalue problem between the Fréchet mean $G_{\mathcal{T}_i}$ and classifier patterns [55] that were back projected to $S^+_D$ with $P_{\mathcal{G}_{\mathcal{T}_i}}^{-1}$. The resulting eigenvectors are the patterns and the eigenvalues $\lambda_k$ reflect the relative source contribution $c_k$:

$$c_k = \max(\lambda_k, \lambda_k^{-1}), \quad \lambda_k = \exp(b_k/||b||_2)$$

(21)

To benefit from the intrinsic interpretability of TSM models, we constrain our hypothesis class $\mathcal{H}$ to functions $h : \mathcal{X} \times I_d \to \mathcal{Y}$ that can be decomposed into a composition of a shared linear feature extractor with covariance pooling $f_\theta : \mathcal{X} \to S^+_D$, domain-specific tangent space mapping $m_\phi : S^+_D \times \mathcal{I}_d \to \mathbb{R}^{D(D+1)/2}$, and a shared linear classifier $g_\psi : \mathbb{R}^{D(D+1)/2} \to \mathcal{Y}$ with parameters $\Theta = \{\theta, \phi, \psi\}$.

*Here, we use the entire dataset’s Fréchet mean instead of the domain-specific ones to compute patterns for the average domain.*
**TSMNet with SPDDSMBN** Unlike previous approaches which learn \( f_\theta, m_\phi, g_\phi \) sequentially \([29,31,33,30]\), we parametrize \( h = g_\phi \circ m_\phi \circ f_\theta \) as a neural network and learn the entire model in an end-to-end fashion (Figure 1). Details of the proposed architecture, denoted TSMNet, are provided in appendix \[A\,2.5\]). In a nutshell, we parametrize \( f_\theta \) as the composition of the first two linear convolutional layers of ShConvNet \([59]\), covariance pooling \([57]\), BiMap \([37]\), and ReEig \([37]\) layers. A BiMap layer applies a linear subspace projection, and a ReEig layer thresholds eigenvalues of symmetric matrices so that the output is SPD. We used the default threshold \((10^{-4})\) and found that it was never active in the trained models. Hence, after training, \( f_\theta \) fulfilled the hypothesis class constraints. In order for \( m_\phi \) to align the domain data and compute TSM, we use SPDDSMBN \([18]\) with shared parameters (i.e., \( G_\phi = G_\phi = I, \nu_\phi = \nu_\phi \)) in \([16]\). Finally, the classifier \( g_\phi \) was parametrized as a linear layer with softmax activations. We use the standard-cross entropy loss as training objective, and optimized the parameters with the Riemannian ADAM optimizer \([58]\).

5 Experiments with EEG data

In the following, we apply our method to classify target labels from short segments of EEG data. We consider two BCI applications, namely, mental imagery \([39,50]\) and mental workload estimation \([60]\). Both applications have high potential to aid society in rehabilitation and healthcare \([61,62,18]\) but have, currently, limited practical value because of poor generalization across sessions and subjects \([19,15]\).

**Datasets and preprocessing** The considered mental imagery datasets were BNCI2014001 \([53]\) (9 subjects/2 sessions/4 classes), BNCI2015001 \([62]\) (12/2-3/2), Lee2019 \([65]\) (54/2/2), Lehner2020 \([66]\) (1/7/2), Stieger2021 \([67]\) (62/4-8/4) and Hehnerberger2021 \([68]\) (1/26/4). For mental workload estimation, we used a recent competition dataset \([69]\) (12/2/3). A detailed description of the datasets is provided in appendix \[A\,2.4\]). Altogether, we analyzed a total of 603 sessions of 158 human subjects whose data was acquired in previous studies that obtained the subjects’ informed consent and the right to share anonymized data.

The python packages moabb \([14]\) and mne \([70]\) were used to preprocess the datasets. The applied steps comprise resampling the EEG signals to 250/256 Hz, applying temporal filters to extract oscillatory EEG activity in the 4 to 36 Hz range (spectrally resolved if required by a method) and finally extract short segments \((\leq 3s)\) associated to a class label (details provided in appendix \[A\,2.2\]).

**Evaluation** We evaluated TSMNet against several baseline methods implementing direct transfer or multi-source (-target) UDA strategies. They can be broadly categorized as component based \([71,68]\), Riemannian geometry aware \([12,17,50,72]\) or deep learning \([56,74,65]\). All models were fit and evaluated with a randomized leave 5% of the sessions (inter-session TL) or subjects (inter-subject TL) out cross-validation (CV) scheme. For inter-session TL, the models were only provided with data of the associated subject. When required, inner train/test splits (neural nets) or CV (shallow methods) were used to optimize hyper parameters (e.g., early stopping, regularization parameters). The dataset Hehnenberger2021 was used to fit the hyper parameters of TSMNet, and is, therefore, omitted in the presented results. Balanced accuracy (i.e., the average recall across classes) was used as scoring metric. As the discriminability of the data varies considerably across subjects, we decided to report the results in the figures relative to the score of a SoA domain-specific Riemannian geometry aware method \([17]\), which was fitted and evaluated in a 80%/20% chronological train/test split (for details see appendix \[A\,2.4\]).

**Soft- and hardware** We either used publicly available python code or implemented the methods in python using the packages torch \([74]\), scikit-learn \([75]\), skorch \([76]\), geoopt \([77]\), mne \([70]\), pyriemann \([78]\), pymanopt \([79]\). We ran the experiments on standard computation PCs equipped with 32 core CPUs with 128 GB of RAM and used up to 1 GPU (24 GRAM). Depending on the dataset size, fitting and evaluating TSMNet varied from a few seconds to minutes.

5.1 Mental imagery

**TSMNet closes the gap to domain-specific methods** Figure 2 summarizes the mental imagery results. It displays test set scores of the considered TL methods relative to the score of a SoA domain-specific reference method. Combining the results of all subjects across datasets (Figure 2a), it becomes apparent that TSMNet is the only method that can significantly reduce the gap to the reference method (inter-subject) or even exceed its performance (inter-session). Figure 2b displays the results resolved across datasets (for details see appendix \[A\,3.1\]). We make two important observations. First, concerning inter-session TL, TSMNet meets or exceeds the score of the reference method.
Figure 2: Mental imagery results (5 datasets). BCI test set score (balanced accuracy) for inter-session/subject transfer learning methods relative to a SoA domain-specific reference model (80%/20% chronological train/test split; for details see appendix A.2.4). a. Barplots summarize the grand average (573 sessions, 138 subjects) results. Errorbars indicate bootstrapped (1e3 repetitions) 95% confidence intervals (over subjects). b. Box and scatter plots summarize the dataset-specific results for selected methods from each category. Datasets are ordered according to the training set size. Each dot summarizes the score for one subject. Lehner2021 is not displayed as it contains only 1 subject.

Figure 3: Model interpretation results. Patterns extracted from a TSMNet. a. Motor imagery dataset (BNCI2015001, inter-subject TL). The top, left panel lists the contribution, defined in (21), for each extracted source $k = 1, \ldots, 20$ (x-axis) to the target class. Panels in the left column summarize the spectral patterns of extracted sources. For visualization purposes, only the 2 most discriminative sources are displayed. Panels in the top row summarize the frequency profile of each spectral channel (output of 4 temporal convolution layers in $f_0$). Topographic plots summarize how the source activity is projected to regions covered by the EEG channels (rows correspond to the source index; columns to spectral channels). EEG channels at darker blue or red areas capture more source activity and are, therefore, more discriminative. b. As in a for the mental workload estimation dataset and class low.

Consistently across datasets. Second, concerning inter-subject TL, we found that all considered methods tend to reduce the performance gap as the dataset size (# subjects) increases, and that TSMNet is consistently the top or among the top methods. As a fitted TSMNet corresponds to a typical TSM model with a linear classifier, we can transform the fitted parameters into interpretable patterns. Figure 3a displays extracted patterns for the BNCI2015001 dataset (inter-subject TL). It is clearly visible that TSMNet infers the target label from neurophysiologically plausible sources (rows in Figure 3a). As expected, the source with highest contribution has spectral peaks in the alpha and beta bands, and originates in contralateral and central sensorimotor cortex.

DSBN on $S_D^+$ drives the success of TSMNet. Since TSMNet combines several advances, we present the results of an ablation study in Table 2. It summarizes the grand average inter-session TL test scores relative to TSMNet with SPDDSMBN for $n = 138$ subjects. We observed three significant effects. The largest effect can be attributed to $S_D^+$, as we observed the largest performance decline if the architecture would be modified to omit the SPD manifold (4.5% with DSBN, 3% w/o DSBN).

We replaced the covariance pooling, BiMap, ReEig, SPD(DS)MBN, LogEig layers with variance pooling, elementwise log activations followed by (DS)MBN. Note that the resulting architecture is similar to ShConvNet.
Table 2: Ablation study. Grand average (5 mental imagery datasets, 138 subjects, inter-session TL) score for the test data relative to the proposed method, and training fit time (50 epochs). Standard-deviation is used to report the variability across subjects. Permutation t-tests (1e4 perms, df=137, 4 tests with t-max adjustment) were used to identify significant effects.

<table>
<thead>
<tr>
<th>$S^+_D$</th>
<th>DSBN [1]</th>
<th>BN method</th>
<th>$\Delta$ balanced accuracy (%) mean (std)</th>
<th>t-val (p-val) mean (std) fit time (s) mean (std)</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>yes</td>
<td>SPDMBN (proposed)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>yes</td>
<td>yes</td>
<td>SPDBN [4]</td>
<td>-1.6 ( 2.2)</td>
<td>-7.8 (0.0001)</td>
</tr>
<tr>
<td>no</td>
<td>yes</td>
<td>SPDMBN (proposed)</td>
<td>-3.9 ( 4.4)</td>
<td>-10.7 (0.0001)</td>
</tr>
<tr>
<td>no</td>
<td>yes</td>
<td>MBN [2]</td>
<td>-4.5 ( 3.8)</td>
<td>-10.1 (0.0001)</td>
</tr>
<tr>
<td>no</td>
<td>yes</td>
<td>MBN [2]</td>
<td>-6.9 ( 4.8)</td>
<td>-13.4 (0.0001)</td>
</tr>
</tbody>
</table>

The performance gain comes at the cost of a 2.6x longer time to fit the parameters. The second largest effect can be attributed to DSBN; without DSBN the performance dropped by 3.9% (with $S^+_D$) and 2.4% (w/o $S^+_D$). The smallest, yet significant effect can be attributed to SPDMBN.

5.2 Mental workload estimation

Compared to the baseline methods, TSMNet obtained the highest average scores of 54.7% (7.3%) and 52.4% (8.8%) in inter-session and -subject TL (for details see appendix A.3.1). Interestingly, the inter-session TL score of TSMNet matches the score (54.3%) of the winning method in last year’s competition [15]. To shed light on the sources utilized by TSMNet, we show patterns for a fitted model in Figure 3b. For the low mental workload class, the top contributing source’s activity peaked in the theta band and originated in pre-frontal areas. The second source’s activity originated in occipital cortex with non-focal spectral profile. Our results agree with the findings of previous research, as both areas and the theta band have been implicated in mind wandering and effort withdrawal [60].

6 Discussion

In this contribution, we proposed a machine learning framework around (domain-specific) momentum batch normalization on $S^+_D$ to learn tangent space mapping (TSM) and feature extractors in an end-to-end fashion. In a theoretical analysis, we provided error bounds for the running estimate of the Fréchet mean as well as convergence guarantees under reasonable assumptions. We then applied the framework, to a multi-source multi-target unsupervised domain adaptation problem, namely, inter-session and -subject transfer learning for EEG data and obtained or attained state-of-the art performance with a simple, intrinsically interpretable model, denoted TSMNet, in a total of 6 diverse BCI datasets (138 human subjects, 573 sessions). In the case of mental imagery, we found that TSMNet significantly reduced (inter-subject TL) or even exceeded (inter-session TL) the performance gap to a SoA domain-specific method.

Although our framework could be readily extended to online UDA for unseen target domains, we limited this study to offline evaluations and leave actual BCI studies to future work. A limitation of our framework, and also any other method that involves eigen decomposition, is the computational complexity, which limits its application to high-dimensional SPD features (e.g., fMRI connectivity matrices with fine spatial granularity). Altogether, the presented results demonstrate the utility of our framework and in particular TSMNet as it not only achieves highly competitive results but is also intrinsically interpretable. While we do not foresee any immediate negative societal impacts, we provide direct contributions towards the scalability and acceptability of EEG-based healthcare [11, 5] and consumer [18, 60] technologies. We expect future works to evaluate the impact of the proposed methods in clinical applications of EEG like sleep staging [60, 81], seizure [82] or pathology detection [83, 84].

Acknowledgments and Disclosure of Funding

This work was supported by the JSPS KAKENHI (Grants-in-Aid for Scientific Research) under Grant Numbers 20H04249, 20H04208, 21H03516 and 21K12055, and the Agency for Medical Research and Development (AMED) under Grant Number JP22dm0307009.
References


Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See section 6
(c) Did you discuss any potential negative societal impacts of your work? [Yes] See section
6.1
(d) Have you read the ethics review guidelines and ensured that your paper conforms to
them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See paragraph
   4 in section 3.
   (b) Did you include complete proofs of all theoretical results? [Yes] See appendix A.1 in
   the supplementary material.

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main exper-
   imental results (either in the supplemental material or as a URL)? [Yes] See Supplementary
   Material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they
   were chosen)? [Yes] See paragraph 2 in section 5 for a brief summary and appendix
   A.2 for details.
   (c) Did you report error bars (e.g., with respect to the random seed after running exper-
   iments multiple times)? [Yes] We provide error bars with respect to the variability
   across sessions and subjects.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type
   of GPUs, internal cluster, or cloud provider)? [Yes] See paragraph 3 in section 5.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See paragraph 1 in
   section 5.
   (b) Did you mention the license of the assets? [Yes] See appendix A.2.1 in the Supplemen-
   tary material.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you’re
   using/curating? [Yes] See paragraph 1 in section 5.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable
   information or offensive content? [Yes] See paragraph 1 in section 5.

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if
   applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review
   Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount
   spent on participant compensation? [N/A]